BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

WEST Search History

DATE: Wednesday, June 19, 2002

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=OR		
L19	L18 and l1	3	L19
L18	L16 and l11	33	L18
L17	L16 and l11 and l14	0	L17
L16	dynamic adj object$1	351	L16
L15	L14 and l11	6	L15
L14	registry same l6	23	L14
L13	L12 and l2	3	L13
L12	L11 and l6	65	L12
L11	component adj object adj model	462	L11
L10	L9 and com	0	L10
L9	l3 and l6	12	L9
L8	l3 and l6L7	0	L8
L7	L6 and l5	0	L7
L6	data adj store	9721	L6
L5	L4 and l1	2	L5
L4	L3 and l2	180	L4
L3	L2 same object$1	180	L3
L2	dynamic adj behavior$1	1845	L2
L1	((709/310 |709/311 |709/312 |709/313 |709/314 |709/315 |709/316 |709/317 |709/318 |709/319 |709/320)!.CCLS.)	1233	L1

END OF SEARCH HISTORY

WEST Search History

DATE: Wednesday, June 19, 2002

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=OR		
L19	L18 and l1	3	L19
L18	L16 and l11	33	L18
L17	L16 and l11 and l14	0	L17
L16	dynamic adj object$1	351	L16
L15	L14 and l11	6	L15
L14	registry same l6	23	L14
L13	L12 and l2	3	L13
L12	L11 and l6	65	L12
L11	component adj object adj model	462	L11
L10	L9 and com	0	L10
L9	l3 and l6	12	L9
L8	l3 and l6L7	0	L8
L7	L6 and l5	0	L7
L6	data adj store	9721	L6
L5	L4 and l1	2	L5
L4	L3 and l2	180	L4
L3	L2 same object$1	180	L3
L2	dynamic adj behavior$1	1845	L2
L1	((709/310 |709/311 |709/312 |709/313 |709/314 |709/315 |709/316 |709/317 |709/318 |709/319 |709/320)!.CCLS.)	1233	L1

END OF SEARCH HISTORY

DO NOT REMOVE THIS SEARCH REPORT FROM FILE!

ST. JOHN COURTENAY III
PRIMARY EXAMINER

WEST

Generate Collection Print

L15: Entry 4 of 6 File: USPT May 1, 2001

US-PAT-NO: 6226692
DOCUMENT-IDENTIFIER: US 6226692 B1

TITLE: Method and system for constructing software components and systems as assemblies of independent parts

DATE-ISSUED: May 1, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Miloushev; Vladimir I.	Laguna Niguel	CA		
Nickolov; Peter A.	Irvine	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Object Dynamics Corporation	Costa Mesa	CA			02

APPL-NO: 9/ 077796 [PALM]
DATE FILED: October 28, 1998

PARENT-CASE:
This application depends for priority upon U.S. Provisional Patent Application Ser. No. 60/008,699, filed Dec. 15, 1995, which is incorportated herein in its entirety by reference thereto.

PCT-DATA:

APPL-NO	DATE-FILED	PUB-NO	PUB-DATE	371-DATE	102(E)-DATE
PCT/US96/19675	December 13, 1996	WO97/22925	Jun 26, 1997	Oct 28, 1998	Oct 28, 1998

INT-CL: [7] G06 F 9/00

US-CL-ISSUED: 709/316
US-CL-CURRENT: 709/316

FIELD-OF-SEARCH: 709/103, 709/223, 709/310-332, 709/316

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5295242	March 1994	Mashruwala et al.	709/223
☐	5481715	January 1996	Hamilton et al.	709/316
☐	5692183	November 1997	Hapner et al.	707/103
☐	5751962	May 1998	Fanshier et al.	709/223
☐	5848419	December 1998	Hapner et al.	707/103

ART-UNIT: 211
PRIMARY-EXAMINER: Courtenay, III; St. John
ATTY-AGENT-FIRM: Wittenberg; Malcolm B. Crosby, Heafey, Roach & May

ABSTRACT:

A system and a method for designing and constructing software components and systems by assembling them from independent parts which is compatible with and extends existing object models. A terminal interface and a terminal mechanism for interfacing objects is included. The mechanism is independent from the actual type of interactions established through it and allows objects to invoke directly services of other objects. All objects in a given system implement and expose a terminal interface. A property interface and mechanism with hierarchical property names and ability to execute queries is also included. The mechanism can be used for parameterization and serialization of objects, as well as to provide structured storage. A new and advantageous type of software object, named parts, is defined. Parts are constructed through an abstract factory and implement a property interface and a terminal interface.

7 Claims, 58 Drawing figures

L15: Entry 4 of 6 File: USPT May 1, 2001

DOCUMENT-IDENTIFIER: US 6226692 B1
TITLE: Method and system for constructing software components and systems as assemblies of independent parts

Brief Summary Paragraph Right (4):
Over the last fifteen years, the object paradigm, including object-oriented analysis, design, programming and testing, has become the predominant paradigm for building software systems. A wide variety of methods, tools and techniques have been developed to support various aspects of object-oriented software construction, from formal methods for analysis and design, through a number of object-oriented languages, component object models and object-oriented databases, to a number of CASE systems and other tools that aim to automate one or more aspects of the development process.

Brief Summary Paragraph Right (12):
Object-oriented software development is based on the the object-model concept, which defines the structure of objects, their attributes and interactions. While many and different object models are in use, they have many commonalties and are well understood and described in the software engineering community. Most of the known object models fall into one of the two broad categories: object-oriented languages and component object models.

Brief Summary Paragraph Right (14):
Component object models advance further the concepts of modularity in object-oriented technologies by defining language-neutral standards for implementation and management of objects as well as for interactions between them. Detailed specifications and architecture definitions for the two most widely accepted component object models as well as many examples and guidelines for building software systems based on them are provided in the following two documents: (1) Microsoft's "The Component Object Model Specification", dated Mar. 6, 1995 and available from Microsoft Corporation; (2) "The Common Object Request Broker Architecture" available from Object Management Group, Inc., Framingham Corporate Center, 492 Old Connecticut Path, Framingham, Mass. 01701.

Brief Summary Paragraph Right (15):
Most object-oriented systems and applications utilize a number of commonly recognized mechanisms, such as separating interfaces from implementations, abstract factories, parameterization, serialization, and structured storage. Abstract interfaces, along with multiple interfaces per object and abstract factories are well explained in the Component Object Model (COM) specification cited above. The abstract factory pattern is also described very well in the book "Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma et al., published by Addison-Wesley Publishing, Reading, Mass., in 1994.

Brief Summary Paragraph Right (17):
Component object models normally define object properties as a linear, flat table of named attributes To represent more sophisticated structures through this model a sequence of operations on properties is needed, such as when setting a value on a given property will modify the mapping of some other properties to a new set of attributes, as it is widely done in Visual Basic. When using such a model for parameterization, the need for sequencing requires either specific coding or setting the properties during parameterization in a specific order, both of which require

custom work for each component.

Brief Summary Paragraph Right (33):
Microsoft Component Object Model (COM) defines a mechanism known as connectable objects as described in the OLE Control Developer's Kit listed above and U.S. Pat. No. 5,485,617, Stutz et al., Jan. 16, 1996, "Method and System for Dynamically-Generating Object Connections", which is incorporated here in its entirety by reference.

Brief Summary Paragraph Right (46):
To gain wide acceptance and become truly usable in development of commercially viable software products, object composition requires methods and tools that make it easy to use within the established and generally accepted object models, including object-oriented languages and component object models. Such methods and tools should work within the typical small scale, or fine granularity, of objects in such models, should not impose additional call-time overhead, should be easy to use in combination with other methods and third-party components, and should not require excessively large investments in tools and education through steep learning curves to practice them.

Detailed Description Paragraph Right (67):
The present invention does not define a new object model. Instead the present invention extends existing object models in a way that provides the necessary mechanisms for considerable improvements over available OOP tools, thereby maintaining compatibility with existing systems to ensure desirability for the establishment software development market. The present invention may be used with the most popular object models now available, including models defined by object-oriented languages, such as C++ and Smalltalk, as well as models defined by component objects systems, such as Microsoft's Component Object Model (COM), IBM's System Object Model (SOM), the Common Object Request Broker Architecture (CORBA) and many others known in the art to which the present invention pertains.

Detailed Description Paragraph Right (215):
When implementing a property mechanism, one must define a base set of property types that the particular implementation of the property mechanism will support. One rule for defining such a set is to represent the basic data types defined by the language in which objects will be implemented. This way the implementation can easily and conveniently map any required attribute of an object to a property. Another rule is to include in the set data types or property types defined by an external system with which the objects will need to interact. This is especially important when building a system that needs to exchange data with objects defined under OLE conventions. Language-based object models generally lack any notion of properties, while component object models assume that their definition of properties and property types is the only one that is valid.

Detailed Description Paragraph Right (243):
The property interface as defined by the present invention and its hierarchical name space can be used as a primary interface to a variety of hierarchical data store implementations, such as registries and data repositories. In essence, the mechanism provides a general purpose flexible model for describing data structures of arbitrary complexity utilizing a simple minimalistic interface.

Detailed Description Paragraph Right (475):
While the present invention has been described with reference to certain preferred embodiments, those skilled in the art to which the present invention pertains will now, as a result of the applicant's teachings herein, recognize that various modifications and other embodiments may be provided. By way of example, the precise structure of the terminal or other interfaces may be modified while still preserving the advantages of the invention. Similarly, the present invention can be easily adapted to a variety of object models, including object-oriented languages, such as C++, and component object models, such as Microsoft COM. Adapting the present invention for use with C++ may take the form of a class or template library, defining a base class PART and a base class ASSEMBLY, that implement the respective behavior as described above; user-defined parts and assemblies can be implemented by deriving from these base classes. Alternatively, the definition of the C++ language

WEST Search History

DATE: Wednesday, June 19, 2002

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L15	L14 and l11	6	L15
L14	registry same l6	23	L14
L13	L12 and l2	3	L13
L12	L11 and l6	65	L12
L11	component adj object adj model	462	L11
L10	L9 and com	0	L10
L9	l3 and l6	12	L9
L8	l3 and l6L7	0	L8
L7	L6 and l5	0	L7
L6	data adj store	9721	L6
L5	L4 and l1	2	L5
L4	L3 and l2	180	L4
L3	L2 same object$1	180	L3
L2	dynamic adj behavior$1	1845	L2
L1	((709/310 |709/311 |709/312 |709/313 |709/314 |709/315 |709/316 |709/317 |709/318 |709/319 |709/320)!.CCLS.)	1233	L1

END OF SEARCH HISTORY

WEST Search History

DATE: Wednesday, June 19, 2002

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=OR		
L15	L14 and l11	6	L15
L14	registry same l6	23	L14
L13	L12 and l2	3	L13
L12	L11 and l6	65	L12
L11	component adj object adj model	462	L11
L10	L9 and com	0	L10
L9	l3 and l6	12	L9
L8	l3 and l6L7	0	L8
L7	L6 and l5	0	L7
L6	data adj store	9721	L6
L5	L4 and l1	2	L5
L4	L3 and l2	180	L4
L3	L2 same object$1	180	L3
L2	dynamic adj behavior$1	1845	L2
L1	((709/310 |709/311 |709/312 |709/313 |709/314 |709/315 |709/316 |709/317 |709/318 |709/319 |709/320)!.CCLS.)	1233	L1

END OF SEARCH HISTORY

WEST

Generate Collection Print

L5: Entry 1 of 2 File: USPT Jul 11, 2000

US-PAT-NO: 6088739
DOCUMENT-IDENTIFIER: US 6088739 A

TITLE: Method and system for dynamic object clustering

DATE-ISSUED: July 11, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Pugh; David	Bellevue	WA		
Ball; John Eugene	Woodinville	WA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Microsoft Corporation	Redmond	WA			02

APPL-NO: 8/ 673443 [PALM]
DATE FILED: June 28, 1996

INT-CL: [7] G06 F 9/44

US-CL-ISSUED: 709/315; 395/500.43
US-CL-CURRENT: 709/315; 703/22

FIELD-OF-SEARCH: 395/500, 395/680, 395/683, 395/500.28-500.49, 364/578, 709/300-305

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5050074	September 1991	Marca	364/200
☐	5481718	January 1996	Ryu et al.	395/683
☐	5710925	January 1998	Leach et al.	709/303
☐	5737606	April 1998	Martin et al.	709/303
☐	5764958	June 1998	Coskun	395/500
☐	5805885	September 1998	Leach et al.	395/683
☐	5874954	February 1999	Kilmer et al.	345/333

OTHER PUBLICATIONS

Howard, Duncan, "An Introduction to MUD, Multi-User Dungeon," Century

Communications, London, 1985.

ART-UNIT: 275
PRIMARY-EXAMINER: Banankhah; Majid A.
ASSISTANT-EXAMINER: Courtenay, III; St. John
ATTY-AGENT-FIRM: Dryja; Michael

ABSTRACT:

A method and system for dynamically modifying the behavior of a statically declared object that represents a simulated entity is provided. In a preferred embodiment, a clustering mechanism is provided that represents each simulated entity as a composite object, which is implemented as a composite object of component objects. The clustering mechanism allows the behavior of the composite object to be dynamically modified by adding interfaces to or removing interfaces from the composite object. Each interface, referred to as a role interface, includes methods that implement the behavior associated with a role that the composite object may assume. The clustering mechanism provides a negotiation procedure for attaching roles to the composite object. The component objects that comprise a composite object include a cluster object and zero or more role objects. The cluster object keeps track of the role objects currently within the composite object and exposes each of the role interfaces of each role object outside the composite object. The cluster object also contains functions for retrieving role interfaces and for invoking a function of an exposed role interface.

14 Claims, 24 Drawing figures

PALM INTRANET

Day : Wednesday
Date: 6/19/2002
Time: 12:29:59

Inventor Information for 09/199604

Inventor Name	City	State/Country
SOBESKI, DAVID A.	REDMOND	WASHINGTON
ANDREW, FELIS G.T.J.	SEATTLE	WASHINGTON

Appln Info | Contents | Petition Info | Atty/Agent Info | Continuity Data | Foreign Data

Search Another: Application# [ ] Search **or Patent#** [ ] Search

PCT / [ ] / [ ] Search or PG PUBS # [ ] Search

Attorney Docket # [ ] Search

Bar Code # [ ] Search

(To Go BACK Use BACK Button on Your BROWSER Tool Bar)

Back to PALM | ASSIGNMENT | OASIS | Home page

WEST

End of Result Set

Generate Collection | Print

L19: Entry 3 of 3 File: USPT May 9, 2000

US-PAT-NO: 6059838
DOCUMENT-IDENTIFIER: US 6059838 A

TITLE: Method and system for licensed design and use of software objects

DATE-ISSUED: May 9, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Fraley; Christopher Lee	Woodinville	WA		
Halcoussis; Michael	Woodinville	WA		
Zimmerman; Christopher Alan	Bellevue	WA		
Carter; Alan W.	Bellevue	WA		
Wiltamuth; Scott Michael	Seattle	WA		
Burd; Gary S.	Kirkland	WA		
Hodges; C. Douglas	Redmond	WA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Microsoft Corporation	Redmond	WA			02

APPL-NO: 9/ 234824 [PALM]
DATE FILED: January 21, 1999

PARENT-CASE:
RELATED APPLICATION DATA This application is a division of application Ser. No. 08/870,171, filed Jun. 6, 1997, the disclosure of which is incorporated by reference.

INT-CL: [7] G06 F 9/445

US-CL-ISSUED: 717/1; 717/2, 717/3, 707/104, 709/303, 380/4
US-CL-CURRENT: 717/108; 705/59, 707/104.1, 709/315

FIELD-OF-SEARCH: 395/701-703, 707/104, 707/103, 707/10, 709/302, 709/303, 380/3, 380/21, 380/59, 380/4

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5463769	October 1995	Tate et al.	709/305
☐	5491820	February 1996	Belove et al.	707/3
☐	5560014	September 1996	Imamura	395/701
☐	5572643	November 1996	Judson	709/218
☐	5652888	July 1997	Burgess	709/303
☐	5740549	April 1998	Reilly et al.	705/14
☐	5761499	June 1998	Sonderegger	707/10
☐	5778227	July 1998	Jordan	709/302
☐	5872974	February 1999	Mezick	395/701
☐	5893118	April 1999	Sonderegger	707/203

OTHER PUBLICATIONS

Karl Dakin, "The Transaction-Based Pricing Trend [for software packages]," IEEE Software, vol. 13; Issue 3, pp. 96-97, May 1996.
Kleinoder et al, "MetaJava: An Efficient Run-Time Meta Architecture for JavaTM," Proceedings of the Fifth International Workshop on Object-Orientation in Operation Systems, pp. 54-61, Oct. 27-28, 1996.
David Chappell, "Understanding ActiveX and OLE," Microsoft Press, pp. 203-235, Sep. 24, 1996.
Al Williams, "Visual Basic 5 and ActiveX Controls," Dr. Dobbs Journal, vol. 22, No. 3, p.74(8), Mar., 1997.
"ActiveX Controls: Licensing an ActiveX Control" Document, 4 pages [online]. Microsoft Corp., [retrieved on Dec. 6, 1999]. Retrieved from Internet: .
"Licensing ActiveX Controls" Document, 8 pages [online.] Microsoft Corp., [retrieved on Dec. 6, 1999]. Retrieved from the Internet: .
ActiveX Designer Programmer's Reference, Microsoft Corporation, 100 pages, 1996.

ART-UNIT: 272
PRIMARY-EXAMINER: Hafiz; Tariq R.
ASSISTANT-EXAMINER: Pam; Tuan Q.
ATTY-AGENT-FIRM: Klarquist Sparkman Campbell Leigh & Whinston, LLP

ABSTRACT:

A componentizing object designer is used to define a componentization of visual forms and other object-oriented technologies. The componentized object designer includes a set of tightly integrated protocols enabling Component Object Model (COM) objects to replace standard built-in visual form and other objects. The componentized object designer allows the design-time object and the run-time object to differ in implementation. The componentized object designer allows class identifiers for the run-time objects which are different than design-time objects. With a different class identifier, the run-time object can be saved as an object which is radically different from the design-time object. This enables the run-time object to be stored in a different object library than the design-time object. The componentized object designer allows for different persistence formats to be saved for run-time objects. The persistence

formats for the run-time objects can be significantly smaller in size compared to the original the design-time objects. This is important when the run-time object needs to be downloaded over a computer network like the Internet or an intranet. Licensing is aided by checking the object designer for licensing data, and embedding a licensing key into the run-time object.

11 Claims, 16 Drawing figures

L19: Entry 1 of 3 File: USPT May 8, 2001

DOCUMENT-IDENTIFIER: US 6230312 B1
TITLE: Automatic detection of per-unit location constraints

Brief Summary Paragraph Right (4):
Various types of modular software, including software designed in an object-oriented framework, can conceivably be distributed throughout a distributed system. Object-oriented programming models, such as the Microsoft Component Object Model ("COM"), define a standard structure of software objects that can be interconnected and collectively assembled into an application (which, being assembled from component objects, is herein referred to as a "component application"). The objects are hosted in an execution environment created by system services, such as the object execution environments provided by COM. This system exposes services for use by component application objects in the form of application programming interfaces ("APIs"), system-provided objects and system-defined object interfaces. Distributed object systems such as Microsoft Corporation's Distributed Component Object Model (DCOM) and the Object Management Group's Common Object Request Broker Architecture (CORBA) provide system services that support execution of distributed applications.

Drawing Description Paragraph Right (4):
FIG. 3 is a block diagram of a Microsoft Component Object Model software component that can be used to implement the present invention.

Drawing Description Paragraph Right (6):
FIG. 5 is a block diagram of the component of FIG. 3 with multiple interfaces specified according to Microsoft's Component Object Model.

Detailed Description Paragraph Right (1):
The present invention is directed toward automatic partitioning of units of an application and distribution of those units. In the illustrated embodiment of the present invention, an application is partitioned into one or more application units for distribution in a distributed computing environment. The COIGN system is one possible refinement of the illustrated ADPS that automatically partitions and distributes applications designed according to the Component Object Model ("COM") of Microsoft Corporation of Redmond, Wash. Briefly described, the COIGN system includes techniques for identifying COM components, measuring communication between COM components, classifying COM components, measuring network behavior, detecting component location constraints, generating optimal distribution schemes, and distributing COM components during run-time.

Detailed Description Paragraph Right (13):
With reference now to FIG. 3, in the COIGN system, the computer 20 (FIG. 2) executes "COIGN," a component-based application that is developed as a package of component objects. COIGN's component objects conform to the Microsoft Component Object Model ("COM") specification (i.e., each is implemented as a "COM Object" 60, alternatively termed a "COM component"). COIGN executes using the COM family of services (COM, Distributed COM ("DCOM"), COM+) of the Microsoft Windows NT Server operating system, but alternatively can be implemented according to other object standards (including the CORBA (Common Object Request Broker Architecture) specification of the Object Management Group) and executed under object services of another operating system.

Detailed Description Paragraph Right (108):
Difficulties in classification by profile arise when application units are dynamic

objects, such as COM components, for example. Component lifetimes are dynamic. A component may be instantiated or deleted at almost any point in program execution. Multiple instances of the same static type of component may exist concurrently. Moreover, separate instances of the same static type of component may have vastly different behavior and communication patterns due to their different usage contexts. For example, a single component in the document processing application, Octarine, is instantiated multiple times in a typical execution. Some instances hold references to operations invoked by menu commands. Some instances hold references to parts of a document including footers, headers, and body. Still other instances hold references to components in dialog boxes or spreadsheet cells. Two components with the same static type and similar communication patterns may need to be placed on separate machines if their sets of communicating partners are significantly different. In applications that are input-driven, user input typically drives the dynamic instantiation of application components. For this reason, component behavior varies tremendously between executions.

Detailed Description Paragraph Right (178):
COM components are dynamic objects. Instantiated during an application's execution, components communicate with the application and each other through dynamically bound interfaces. A component frees itself from memory after all references to it have been released by the application and other components. COIGN is particularly aware of component instantiations. Applications instantiate COM components by calling API functions exported from a user-mode COM DLL. Applications bind to the COM DLL either statically or dynamically.

Current US Cross Reference Classification (3):
709/312